Laboratorium grafiki komputerowej i animacji

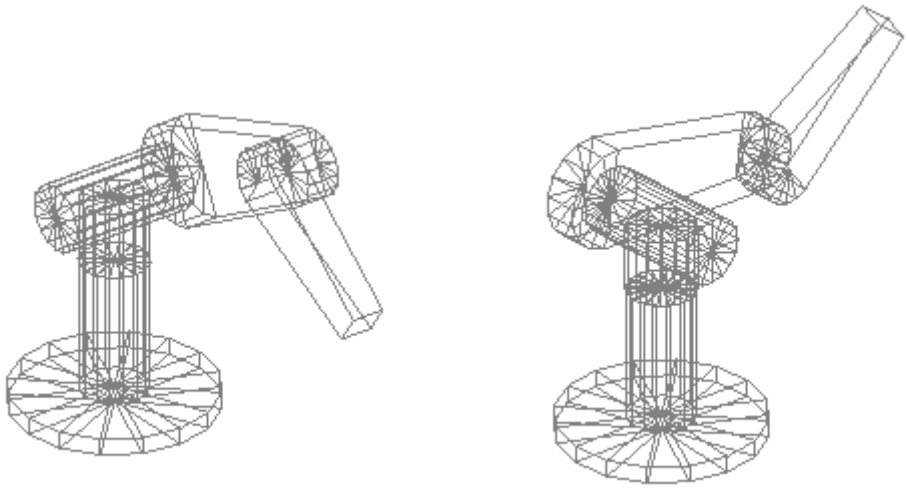
Ćwiczenie III - Biblioteka OpenGL - wprowadzenie, obiekty trójwymiarowe: punkty, linie, wielokąty

Przygotowanie do ćwiczenia:

1. Zapoznać się z ogólną charakterystyką biblioteki OpenGL.
2. Zapoznać się ze sposobem konstruowania prymitywów graficznych OpenGL.
3. Zapoznać się z zasadami tworzenia aplikacji OpenGL na platformie Windows - zestaw funkcji "wgl", biblioteka GLAUX.
4. Zapoznać się z przykładowymi programami dostarczonymi z wprowadzeniem do ćwiczenia.

Przebieg ćwiczenia:

1. Założenia:
    a. Celem prac na kilku następnych zajęciach jest skonstruowanie modelu graficznego manipulatora Puma (Siatka całego manipulatora pokazna jest na rysunku 1.1.).
    b. **Przedmiotem obecnych zajęć jest przygotowanie siatek elementów składowych manipulatora: walca i dwu ramion**.
    c. Wynikiem prac na dzisiejszych zajęciach ma być program zbliżony w działaniu do programu „**puma_elementy.exe**” dostarczonego do materiałów laboratoryjnych.
    d. Realizacja ćwiczenia polega na modyfikacji kodu programu „**gl_template**” dołączonego do materiałów laboratoryjnych.
    e. W realizacji prac wzorować się należy na rozwiązaniach przyjętych w programie „**triangle**” również dołączonym do materiałów laboratoryjnych.



Rysunek 1.1 Graficzny model manipulatora Puma (siatka)

2. Uwagi do sposobu realizacji celu dzisiejszych zajęć laboratoryjnych:
    a. W programie „gl_template” wbudowano możliwość obracania tworzonych modeli przy pomocy klawiszy strzałek.

b. Obiekty graficzne należy tworzyć w funkcji RenderScene() programu „gl_template”.
c. Przy tworzeniu zamkniętych brył kluczową rolę odgrywa takie zaprojektowanie wielokątów, aby były zwrócone na zewnątrz „zewnętrznymi powierzchniami”. Umożliwia to później przyspieszenie obliczeń sceny przez odrzucenie „wewnętrznych” powierzchni w obliczeniach. Upraszcza się wówczas również procedura oświetlania obiektów. W związku z tym podczas realizacji ćwiczenia należy zwrócić szczególną uwagę na odpowiednie dobieranie kolejności podawania wierzchołków wielokątów. Pomocne okazać się może wywołanie funkcji:
glPolygonMode(GL_BACK,GL_LINE);
umieszczone w funkcji RenderScene(). „Wewnętrzne” ściany wielokątów będą wtedy rysowane w postaci siatki, natomiast „zewnętrzne” zostaną w całości zamalowane.
Wywołanie funkcji glPolygonMode(GL_FRONT_AND_BACK,GL_LINE) umożliwi postrzeganie zaprojektowanej bryły w postaci siatki.
d. Każdy z tworzonych na scenie obiektów ma powiązany ze sobą układ współrzędnych. Modele brył tworzyć należy w taki sposób, aby zdawać sobie sprawę, gdzie znajduje się układ współrzędnych powiązany z bryłą (Początkowy bazowy układ współrzędnych sceny znajduje się na środku okna i z tym układem współrzędnych należy wiązać kolejno projektowane siatki brył).
e. Brył zamkniętych nie należy tworzyć z prymitywów do odwzorowywania linii.
f. Wszystkie tworzone w ramach ćwiczeń modele stanowić mają bryły zamknięte, w związku z tym należy precyzyjnie uwzględnić każdą ze ścian bryły (walec powinien zawierać obie podstawy, wielościany powinny posiadać zdefiniowane wszystkie ściany najlepiej w postaci osobnych wielokątów GL_QUADS, GL_TRANGLES, GL_POLYGON).
g. Każdy z elementów manipulatora najwygodniej zamknąć w jednej funkcji. W parametrach wywołania funkcji nie należy uwzględniać położenia elementu na scenie (Sposób rozmieszczania elementów na scenie określony zostanie na kolejnych zajęciach).
h. Sposób uzupełnienia menu na potrzeby programu można wzorować na programie „triangle”.
i. Model walca na potrzeby programu najwygodniej zdefiniować w taki sposób, aby można było zadawać jego promień i wysokość, np.: void walec(double h, double r).
j. Fragment programu zawierający definicję części walca może mieć następującą postać:

```
void walec(double h, double r)
{
double angle,x,y;

    glBegin(GL_TRIANGLE_FAN);
    glVertex3d(0.0f, 0.0f, 0.0f);
    for(angle = 0.0f; angle <= (2.0f*GL_PI); angle += (GL_PI/8.0f))
            {
            x = r*sin(angle);
            y = r*cos(angle);
            glVertex3d(x, y, 0.0);
```

```
            }
    glEnd();

    glBegin(GL_QUAD_STRIP);

    for(angle = 0.0f; angle >= -(2.0f*GL_PI); angle -= (GL_PI/8.0f))
            {
            x = r*sin(angle);
            y = r*cos(angle);
            glVertex3d(x, y, h);
            glVertex3d(x, y, 0);
            }
    glEnd();
}
```